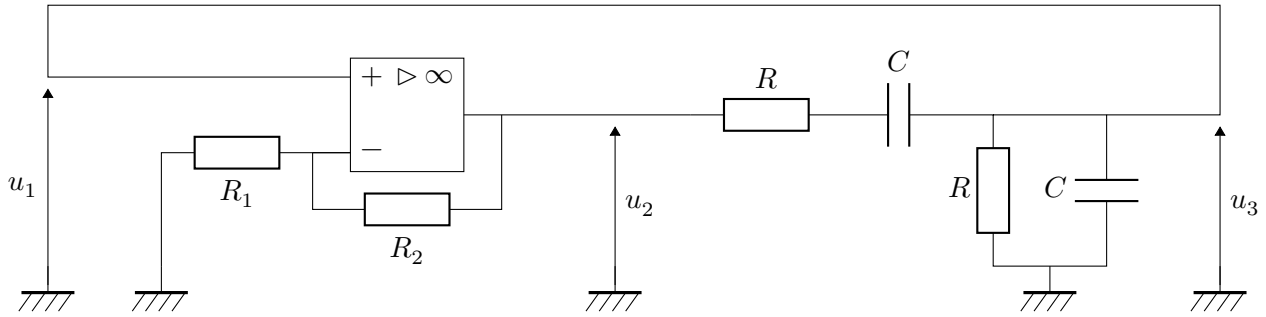
Électronique – Chapitre 3

TP : Simulation d'un signal généré par un oscillateur de Wien

I. Description du problème

On souhaite déterminer l'évolution temporelle d'un signal généré par un oscillateur de Wien, représenté ci-dessous.



En posant $A = 1 + \dfrac{R_2}{R_1}$ le gain de l'amplificateur et $\omega_0 = \dfrac{1}{RC}$ la pulsation propre du filtre, on peut montrer que la tension $u_1(t)$ vérifie une équation différentielle qui dépend du régime de fonctionnement de l'ALI :

- lorsque l'ALI fonctionne en régime linéaire, c'est-à-dire que : $|u_2(t)| < V_{\text{sat}}$, et donc que : $|u_1(t)| < \dfrac{V_{\text{sat}}}{A}$, l'équation différentielle vérifiée est :

$$\frac{\mathrm{d}^2 u_1}{\mathrm{d}t^2} + (3 - A)\,\omega_0 \frac{\mathrm{d}u_1}{\mathrm{d}t} + \omega_0^2\, u_1(t) = 0$$

- lorsque l'ALI fonctionne en régime saturé, c'est-à-dire que : $|u_2(t)| = V_{\text{sat}}$, et donc que : $|u_1(t)| \geqslant \dfrac{V_{\text{sat}}}{A}$, l'équation différentielle vérifiée est :

$$\frac{\mathrm{d}^2 u_1}{\mathrm{d}t^2} + 3\,\omega_0 \frac{\mathrm{d}u_1}{\mathrm{d}t} + \omega_0^2\, u_1(t) = 0.$$

À partir d'une condition initiale quelconque, il est difficile d'obtenir une solution analytique pour $u_1(t)$ comme le régime de fonctionnement de l'ALI, et donc l'équation différentielle régissant le système, peuvent changer au cours du temps. Nous allons donc simuler l'évolution temporelle du signal généré par l'oscillateur à l'aide d'une méthode d'Euler explicite sous Python.

II. Méthode d'Euler appliquée à une équation différentielle du second ordre

II.1. Système de deux équations différentielles du premier ordre

La méthode d'Euler ne permet pas de résoudre directement une équation différentielle du second ordre, mais elle peut résoudre un système de deux équations différentielles du premier ordre. Nous allons alors nous ramener à un tel système en définissant la nouvelle fonction :

$$v_1(t) = \frac{\mathrm{d}u_1}{\mathrm{d}t}.$$

- Lorsque l'ALI fonctionne en régime linéaire, l'équation différentielle du second ordre peut alors s'écrire sous la forme du système :

$$\begin{cases} \dfrac{\mathrm{d}u_1}{\mathrm{d}t} = v_1(t) \\ \\ \dfrac{\mathrm{d}v_1}{\mathrm{d}t} + (3-A)\,\omega_0\,v_1(t) + \omega_0^2\,u_1(t) = 0 \end{cases}$$

- Lorsque l'ALI fonctionne en régime saturé, l'équation différentielle du second ordre peut alors s'écrire sous la forme du système :

$$\begin{cases} \dfrac{\mathrm{d}u_1}{\mathrm{d}t} = v_1(t) \\ \\ \dfrac{\mathrm{d}v_1}{\mathrm{d}t} + 3\,\omega_0\,v_1(t) + \omega_0^2\,u_1(t) = 0 \end{cases}$$

II.2. Discrétisation du temps

Afin de déterminer l'évolution temporelle de $u_1(t)$ et $v_1(t)$, nous allons discrétiser le temps avec un pas temporel δt. Ces fonctions seront alors approximées par des listes $u_1[i]$ et $v_1[i]$ telles que :

$$\begin{cases} u_1[i] = u_1(t_i = i \times \delta t) \\ \\ v_1[i] = v_1(t_i = i \times \delta t) \end{cases}$$

II.3. Approximation des dérivées par la méthode d'Euler explicite

La méthode d'Euler explicite consiste à approximer chaque dérivée à l'aide d'un développement de Taylor au premier ordre en prenant en compte la discrétisation.

Pour exprimer la dérivée première de la tension $u_1(t)$, on réalise le développement de Taylor au premier ordre :

$$u_1(t_i + \delta t) \approx u_1(t_i) + \frac{\mathrm{d}u_1}{\mathrm{d}t}\,\delta t$$

d'où :

$$\frac{\mathrm{d}u_1}{\mathrm{d}t} = \frac{u_1(t_i + \delta t) - u_1(t_i)}{\delta t}$$

avec $u_1(t_i) = u_1[i]$ et $u_1(t_i + \delta t) = u_1[i+1]$ en prenant en compte la discrétisation, et on obtient finalement :

$$\frac{\mathrm{d}u_1}{\mathrm{d}t} = \frac{u_1[i+1] - u_1[i]}{\delta t}.$$

De façon équivalente, la dérivée première de la fonction $v_1(t)$ peut s'exprimer :

$$\frac{\mathrm{d}v_1}{\mathrm{d}t} = \frac{v_1[i+1] - v_1[i]}{\delta t}.$$

II.4. Schéma numérique

Lorsque l'ALI fonctionne en régime linéaire, le système d'équations différentielles du premier ordre se ramène finalement au schéma numérique suivant :

$$\begin{cases} \dfrac{u_1[i+1] - u_1[i]}{\delta t} = v_1[i] \\ \\ \dfrac{v_1[i+1] - v_1[i]}{\delta t} + (3-A)\,\omega_0\,v_1[i] + \omega_0^2\,u_1[i] = 0 \end{cases}$$

d'où les relations de récurrence :

$$\begin{cases} u_1[i+1] = u_1[i] + \delta t\,v_1[i] \\ \\ v_1[i+1] = [1 - (3-A)\,\omega_0\,\delta t]\,v_1[i] - \omega_0^2\,\delta t\,u_1[i] \end{cases}$$

Application :

Lorsque l'ALI fonctionne en régime saturé, montrons qu'à partir du système d'équations différentielles du premier ordre, on obtient les relations de récurrence :

$$\begin{cases} u_1[i+1] = u_1[i] + \delta t\, v_1[i] \\ \\ v_1[i+1] = (1 - 3\,\omega_0\,\delta t)\, v_1[i] - \omega_0^2\,\delta t\, u_1[i] \end{cases}$$

III. Algorithme de résolution numérique

III.1. Génération d'oscillations quasi-sinusoïdales

On considère un oscillateur de Wien dont les valeurs des composants sont : $R_1 = 10\,\text{k}\Omega$, $R_2 = 21\,\text{k}\Omega$, $R = 1\,\text{k}\Omega$ et $C = 1\,\mu\text{F}$, et dont la tension de saturation de l'ALI idéal est : $V_{\text{sat}} = 13\,\text{V}$. À partir de la condition initiale $u_1(t=0) = 0{,}1\,\text{V}$ et $v_1(t=0) = 0$, on souhaite observer l'évolution temporelle de la tension $u_1(t)$ qui doit tendre vers des oscillations quasi-sinusoïdales puisque la valeur du gain de l'amplificateur est légèrement supérieure à 3.

Application :

En utilisant la bibliothèque `matplotlib.pyplot as plt`, écrivons un programme Python permettant de visualiser l'évolution temporelle de la tension $u_1(t)$ en réalisant les étapes suivantes :

- définir les valeurs des composants R_1, R_2, R et C ;
- définir la valeur de la tension de saturation V_{sat} ;
- définir le pas temporel $\delta t = 10\,\mu\text{s}$ et le nombre d'itération $N = 10\,000$;
- calculer le gain de l'amplificateur A et la pulsation propre du filtre ω_0 à partir de leurs expressions ;
- créer la liste t de taille N telle que : $t[i] = i \times \delta t$;
- créer les listes u_1 et v_1 de taille N, initialement remplie de 0 ;
- remplir les premières valeurs $u_1[0]$ et $v_1[0]$ de ces listes afin de prendre en compte la condition initiale $u_1(t=0) = 0{,}1\,\text{V}$ et $v_1(t=0) = 0$;
- pour chaque instant i, calculer les valeurs $u_1[i+1]$ et $v_1[i+1]$ de l'instant suivant en utilisant les relations de récurrence correspondant au régime de fonctionnement de l'ALI ;
- représenter l'évolution temporelle de la tension $u_1(t)$ sur une figure.

III.2. Démarrage des oscillations et oscillations anharmoniques

En partant d'un oscillateur initialement au repos, on rappelle que le démarrage des oscillations n'a lieu que si la valeur du gain de l'amplificateur A est supérieure à 3.

Application :

En simulant l'évolution temporelle de la tension $u_1(t)$ avec des valeurs de R_2 telles que $A < 3$, retrouvons que les oscillations ne démarrent pas.

Si la valeur du gain de l'amplificateur est très supérieure à 3, l'ALI sature souvent au cours d'un cycle, et les oscillations observées ne sont plus sinusoïdales.

Application :

En simulant l'évolution temporelle de la tension $u_1(t)$ avec des valeurs de R_2 telles que $A \gg 3$, retrouvons que des oscillations anharmoniques apparaissent.

III.3. Période des oscillations

La période des oscillations harmoniques générées pour un gain de l'amplificateur $A = 3$ est $T_0 = \dfrac{2\,\pi}{\omega_0}$. Lorsque la valeur du gain augmente, la période T des oscillations anharmoniques générées augmente également, et on souhaite représenter son évolution.

Pour cela, on pourra utiliser la fonction `Periode(t,u)` ci-dessous, prenant en entrée des listes `t` et `u` correspondant à une tension $u(t)$, et qui renvoie en sortie la période T en régime permanent de cette tension si celle-ci oscille autour de 0.

```
def Periode(t,u):
    i=len(t)-1
    while u[i-1]*u[i]>0 :
        i=i-1
    t2=t[i]
    i=i-1
    while u[i-1]*u[i]>0 :
        i=i-1
    t1=t[i]
    return 2*(t2-t1)
```

Application :

En utilisant la bibliothèque `math`, tester la fonction `Periode(t,u)` en l'appliquant à une tension sinusoïdale de la forme :

$$u(t) = \cos\left(\frac{2\,\pi}{T}\,t\right)$$

avec une période $T = 30$ par exemple.
En utilisant la bibliothèque `matplotlib.pyplot as plt`, écrivons un programme Python permettant de représenter l'évolution de la période T des oscillations générées en fonction du gain de l'amplificateur A. On considèrera des oscillateurs dont la résistance R_2 varie de 21 kΩ à 500 kΩ avec un pas de 10 kΩ.
Pour obtenir une évolution moins « bruitée », qui est due à l'erreur de mesure sur la période en régime permanent, on pourra réduire le pas temporel à $\delta t = 1$ µs et augmenter le nombre d'itération à $N = 100\,000$ dans un second temps (attention au temps de calcul de l'ordinateur qui s'allonge fortement).